บุญเลิศ เอี่ยมทัศนา

ซอฟต์แวร์ชิพ สกุล TextEdit

ออบเจ็คของปาสคาล
กับการจัดการกับสตริง

บทความตอนนี้ของผมจะเป็นตอนต่อของการดำเนินการกับสตริง สำหรับท่านที่ไม่ได้ติดตามบทความชุดนี้มาก่อนสามารถที่จะอ่านต่อได้ครับ เพราะผมจะทบทวนเรื่องเก่าพอสมควร

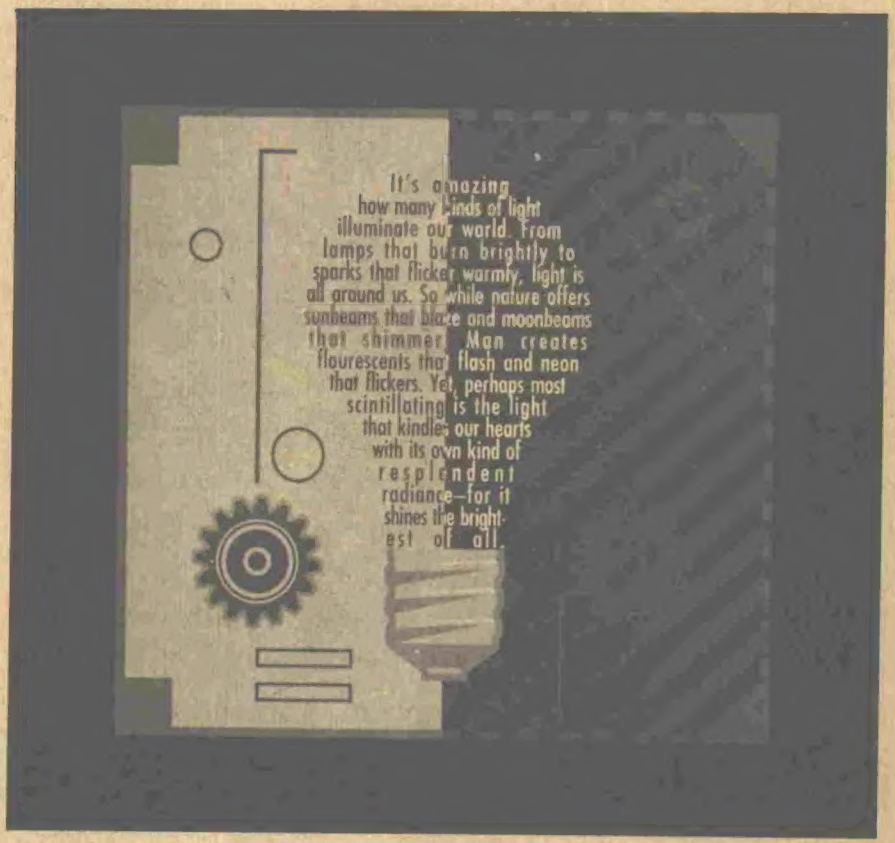
แบ่งมอบงาน

ตามที่ผมเคยกล่าวแล้วว่า ผมต้องการจะแสดงว่าเราจะเขียนโปรแกรมแยกกันเป็นสองกลุ่ม แล้วนำมาต่อกันสมือนกับการต่อวงจรอิเล็กทรอนิกส์ได้อย่างไร โดยอาศัยความสามารถของภาษาที่ให้เขียนโปรแกรมในแบบโอโอพีได้

จึงได้ยกโปรแกรมเอดิเตอร์มาเป็นตัวอย่าง โดยให้กลุ่มหนึ่งดำเนินการเกี่ยวกับคีย์บอร์ดและจอภาพ อีกกลุ่มหนึ่งดำเนินการเกี่ยวกับการเรียบเรียงข้อความที่ผู้ใช้ป้อนในหน่วยความจำ และก็เขียนโปรแกรมในแต่ละกลุ่มเรื่อยมา มาถึงตรงนี้ก็ถึงจุดที่จะต้องแบ่งมอบงานให้เห็นเด่นชัด กล่าวคือ

แนวความคิด

โปรแกรมทั้งสองกลุ่มนี้ผมได้ตั้งชื่อไว้เพื่ออ้างอิงคือ กลุ่มที่ทำงานเกี่ยวกับคีย์บอร์ดและจอภาพให้ชื่อว่า วินโดว์อีดิต อีกกลุ่มหนึ่งให้ชื่อว่า เท็กซ์อีดิต งานหลักของวินโดว์อีดิตคือ ทำหน้าที่เป็นไลน์เอดิเตอร์ (Line editor)



แต่ผมมีแนวความคิดว่าจะให้เท็กซ์อีดิตรับส่งข้อมูลกับวินโดว์อีดิตครั้งละ 1 หน้าต่าง โดยตั้งใจจะเรียกเป็นบอร์ด (board) ซึ่งหมายถึงแผ่นกระดานเพื่อให้มีความหมายในตัว แต่อักษรใกล้เคียงกับบัฟฟ์ (buff) มาก เกรงว่าจะสับสน จึงเปลี่ยนเป็นเพลต (plate)

ความหมายของผมคือ ข้อมูลซึ่งก็คือข้อความที่ผู้ใช้ป้อน 1 เพลต จะมีจำนวนบรรทัดเท่ากับความสูงของหน้าต่าง ในบรรทัดหนึ่ง ๆ จะบรรจุอักษรได้ไม่เกิน 254 ตัว ทำให้วินโดว์อีดิตมีลักษณะเป็น screen editor กลาย ๆ คือในลักษณะที่ขึงตายตัวเหมือนแผ่นกระดานเลื่อนหน้าขึ้นลงไม่ได้ คงเลื่อนได้เฉพาะซ้ายขวาเพื่อให้เห็นอักษรทั้ง 254 คอลัมน์เท่านั้น

เมื่อรวมกับเท็กซ์อีดิตแล้วจะสามารถเลื่อนขึ้นลงได้ ทำให้หน้าต่างมีลักษณะเป็นช่วงสี่เหลี่ยมที่เคลื่อนไปมาบนกระดาษต่อเนื่อง (buff) ที่มีความกว้างได้ถึง 254 อักษร และความยาวไม่จำกัดจำนวนบรรทัด (จนเต็มหน่วยความจำ)

วินโดว์อีดิต

ดังที่กล่าวมาแล้วว่างานหลักของวินโดว์อีดิตคือ ทำหน้าที่เป็นไลน์เอดิเตอร์ แต่มีงานอื่นที่มอบให้อีกคือ

- นำข้อความในเพลตที่เท็กซ์อีดิตส่งมา ออกแสดงทางจอภาพ
- งานไลน์เอดิเตอร์ หมายถึง การอ่านคีย์จากคีย์บอร์ดนำมาตรวจ หากเป็นคีย์อักษร (คือไม่ใช่คีย์ควบคุมและคีย์หน้าที่พิเศษ) ให้นำออกจอภาพและนำไปเก็บในเพลต โดยสรุปคือ เช่นเดียวกับการใช้งานเอดิเตอร์ของเทอร์โบปาสคาลในแต่ละบรรทัด เว้นการทำงานเป็นบล็อก
- งานวินโดว์อีดิต ให้ผู้ใช้สามารถเลื่อนเคอร์เซอร์ขึ้นลง และเมื่อรวมกับไลน์อีดิตจะทำให้สามารถเลื่อนไปได้ทั่วหน้าต่างเพื่อการป้อนและแก้ข้อความ แต่จะเกินเพลตไม่ได้และเมื่อผู้ใช้แทรกหรือลบบรรทัดรวมทั้งการทำงานเป็นบล็อกจะต้องส่งให้เท็กซ์อีดิตเป็นผู้ดำเนินการ
- แปลคีย์ควบคุมและคีย์หน้าที่พิเศษเป็นรหัสคำสั่ง
- เลื่อนขยายและลดขนาดของหน้าต่าง ตามที่ผู้ใช้ต้องการ

เท็กซ์อีดิต

งานหลักของเท็กซ์อีดิตคือ การเรียบเรียงข้อความในหน่วยความจำกับงานอื่นอีก ดังนี้คือ

- เตรียมเพลตให้วินโดว์อีดิต ซึ่งหมายรวมถึงการเลื่อนบรรทัด เลื่อนหน้าต่างขึ้นลง และเลื่อนหน้าต่างไปยังที่ต่าง ๆ ด้วย
- รับเพลตมาเก็บในบัฟฟ์เพื่อการเรียบเรียง
- การเรียบเรียงข้อความในหน่วยความจำยีพได้แก่ การทำงานเป็นบล็อกและเป็นบรรทัด
- การติดต่อกับไฟล์เพื่อการโหลดและเซฟไฟล์ในแบบต่าง ๆ
- งานพิมพ์ออกทางเครื่องพิมพ์
- การค้นหาและเปลี่ยนข้อความ
- ดำเนินการเกี่ยวกับข้อผิดพลาด (error)

คำที่ใช้

เพื่อให้บทความกระชับผมขอเรียนถึงคำที่ผมจะใช้ก่อนครับคือ ข้อความที่ผู้ใช้ป้อนนี้หากอยู่ที่โปรแกรมภายนอกผมเรียกว่า เท็กซ์ หากอยู่ในเอดิเตอร์เรียกว่าบัฟฟ์และเก็บอยู่กับตัวแปร buff ข้อความที่จะนำออกแสดงทางจอภาพจะถูกถ่ายทอดให้แก่ตัวแปร plate และผมเรียกว่า เพลต

ข้อมูลในบัฟฟ์จะจัดเก็บเป็นบรรทัดเรียกว่า ไลน์ (ตัวแปรมีอักษร lne ประกอบ) ไม่จำกัดจำนวนไลน์ ความยาวของไลน์หนึ่ง ๆ ขึ้นอยู่กับความยาวของสตริงที่เป็นจริงจึงไม่จำเป็นต้องเท่ากัน สำหรับตำแหน่งของอักษรในบรรทัดเรียกเป็นอักขระที่ (chn) ซึ่งจะอ้างถึงเป็นไลน์อักขระ หรือ lne-chn

ข้อมูลในเพลตเก็บเป็นบรรทัดเช่นกันเรียกว่า โรว์ (row) จำนวนโรว์เท่ากับความสูงของหน้าต่าง ความยาวของโรว์หนึ่ง ๆ ตายตัวคือ 254 อักขระ ตำแหน่งของอักขระในโรว์เรียกเป็นคอลัมน์ (col) จึงอ้างถึงเป็นโรว์คอลัมน์ หรือ row-col

ตำแหน่งของเคอร์เซอร์บนจอภาพจะตรงกับบนเพลตจึงเรียกเป็นตำแหน่งเคอร์เซอร์ด้วย และยังใช้กับบัฟฟ์ด้วย แต่มีค่าเป็นไลน์อักขระที่ตรงกัน ตัวแปรที่ใช้เก็บตำแหน่งของเคอร์เซอร์ในบัฟฟ์ คือ dstlne กับ dstchn ส่วนของเพลต คือ currow กับ curcol

การถ่ายทอดข้อมูล

การถ่ายทอดข้อมูลจากบัฟฟ์ไป ให้เพลตจะถ่ายทอดตรงช่วงที่จะนำออก จอภาพ โดยลอกข้อมูลจากไลน์ไปให้ โรว์ทีละไลน์ แล้วยกเลิกหน่วยความจำ ที่เก็บข้อมูลของไลน์นั้น นั่นคือ เมื่อ ถ่ายทอดข้อมูลให้เพลตแล้ว ข้อมูลใน บัฟฟ์จะขาดหายไปเท่ากับความสูงของ หน้าต่างเป็นช่องว่างอยู่

เมื่อกลับมาใหม่ก็จะถ่ายทอดข้อมูล ในเพลตทั้งหมดให้บัฟฟ์ โดยจองหน่วย ความจำในฮีพเท่ากับความยาวของสตริง ของโรว์นั้นที่เป็นจริงบวกด้วยหนึ่งเป็น ไลน์ ลอกข้อมูลจากโรว์ให้ไลน์แล้ว เคลียร์คือ ให้ค่า space แก่โรว์นั้นทั้ง โรว์นั่นคือ รับข้อมูลจากเพลตแล้วบัฟฟ์ จะเต็มตามเดิม

การทำงานของเอดิเตอร์โดย ส่วนรวม

จากการที่ผมได้แบ่งโปรแกรมออก เป็น 2 กลุ่มนี้ เมื่อนำมารวมกันจะต้อง จัดให้มีการประสานการติดต่อ โดยจะ จัดให้เอดิเตอร์เป็นผู้ดำเนินการ จึงขอ กล่าวถึงการทำงานของเอดิเตอร์โดย ส่วนรวมก่อนคือ

เมื่อรวมโปรแกรมกันแล้ว การใช้ งานต่อไปจะรู้จักในนามของเอดิเตอร์ เท่านั้น โปรแกรมที่เรียกใช้เอดิเตอร์ สมมติว่าเป็นคอมพายเลอร์ (ของภาษา ใดก็ได้) จะไม่รู้เรื่องราวภายในของ เอดิเตอร์เลยคือ ไม่รู้จักโปรแกรมย่อย

สองกลุ่มดังกล่าวมา

เพื่อให้เห็นภาพเด่นชัดผมขอเริ่มจากผู้ใช้ (ขอให้นึกเปรียบเทียบกับการใช้งานเทอร์โบปาสคาลครับ) เมื่อผู้ใช้รันโปรแกรมคอมพายเลอร์ (รัน TURBO) คอมพายเลอร์จะเรียกใช้เอดิเตอร์และเข้าสู่โหมดการอีดิตคือ ให้ผู้ใช้ป้อนข้อความได้เลย แต่ผู้ใช้อาจเลือกกลับมาที่เมนูหลักเพื่อเลือกรายการเกี่ยวกับไฟล์ คอมพายล์ รัน และอื่น ๆ ซึ่งเอดิเตอร์จะต้องประสานงานกับคอมพายเลอร์ดำเนินการต่อไป

หากย้อนกลับมาทางด้านโปรแกรมบ้าง เมื่อผู้ใช้รันโปรแกรมคอมพายเลอร์ คอมพายเลอร์จะต้องเตรียม (Init) เอดิเตอร์ ซึ่งผมได้กำหนดไว้ว่าจะต้องส่งค่าพารามิเตอร์ต่อไปนี้ให้แก่เอดิเตอร์คือ

- จำนวนบรรทัดของข้อมูล (ขยายได้โดยอัตโนมัติ)
- จุดโคออร์ดิเนตของหน้าต่าง
- ชื่อหน้าต่าง
- แอดเดรสของเมนูหลัก

แล้วสั่งใช้งานเอดิเตอร์ (Action) โดยผ่านค่าพารามิเตอร์ต่อไปนี้ไปให้

- ตัวแปรพอยน์เตอร์เพื่อขอรับเท็กซ์กลับมา
- หมายเลขบรรทัดและอักขระที่ที่ต้องการให้เอดิเตอร์แสดงข้อความส่วนนี้ เช่น เมื่อตรวจพบที่ผิด สำหรับครั้งแรกให้หมายเลขบรรทัดเป็น 1 และอักขระที่เป็น 1
- ตัวแปรบูลีน เพื่อแจ้งให้ทราบว่าพบที่ผิดหรือไม่ ครั้งแรกจะให้เป็นจริง เมื่อประกอบกับหมายเลขของบรรทัดแสดงว่ามีที่ผิดที่บรรทัดที่ 1 อักขระที่ 1 เท่ากับให้แสดงภาพตั้งแต่อักขระตัวแรก
- ในเที่ยวกลับให้รับสถานะว่าผู้ใช้เปลี่ยนแปลงเท็กซ์ ซึ่งจะต้องคอมพายล์ใหม่หรือไม่
- ตัวแปรสตริงเพื่อส่งข้อความเกี่ยวกับข้อผิดพลาดไปออกจอภาพ และขอรับรายการที่ผู้ใช้เลือก เช่น ให้คอมพายล์ ให้รัน เป็นต้น

การติดต่อระหว่างเอดิเตอร์กับคอมพายเลอร์มีเพียงเท่านี้ อนึ่งคือ แม้ผู้ใช้จะเปิดหน้าต่างเพื่อการอีดิตหลายบานหลายไฟล์ คอมพายเลอร์จะไม่ทราบและไม่จำเป็นต้องทราบ คงทำหน้าที่คอมพายล์และรันโปรแกรมตามเท็กซ์ที่ส่งไปให้เท่านั้น อีกเรื่องหนึ่งคือการลบจอภาพทั้งหมด (หากต้องการ) เมื่อเริ่มต้นผมให้เป็นหน้าที่ของคอมพายเลอร์

งานภายในของเอดิเตอร์

ทางด้านการทำงานภายในของเอดิเตอร์ผมได้วางแนวความคิดไว้ว่าเมื่อคอมพายเลอร์สั่งเตรียมเอดิเตอร์ (Init) เอดิเตอร์จะให้เตรียมเท็กซ์อีดิตก่อนเพื่อให้เตรียมบัฟฟ์และเพลตให้ แล้วจึงสั่งเตรียมวินโดว์อีดิตซึ่งวินโดว์อีดิตจะนำเพลตออกแสดงด้วย แต่เนื่องจากบัฟฟ์ว่างทำให้เพลตว่างไม่ปรากฏภาพใดบนจอภาพ เป็นการลบจอภาพในหน้าต่างนี้ไปในตัว

เมื่อคอมพายเลอร์ให้ปฏิบัติ (Action) เอดิเตอร์จะให้ทำงานในวินโดว์อีดิตก่อน ซึ่งผู้ใช้จะเลือกป้อนข้อความหรือเลือกรายการใดก็ได้ จนกว่าผู้ใช้จะเลือกกลับไปคอมพายเลอร์

ในด้านการป้อนข้อความจะคงทำงานอยู่ในวินโดว์อีดิตจนกว่าจะออกไปนอกเพลต เช่น ให้เลื่อนหน้า หรือมีการทำงานเป็นบล็อก เป็นบรรทัด หรือเปลี่ยนขนาดของหน้าต่าง เป็นต้น จึงไปทำงานในเท็กซ์อีดิต ซึ่งเท็กซ์อีดิตจะทราบว่าผู้ใช้ต้องการให้ดำเนินการใดได้จากรหัสคำสั่ง (ดังจะกล่าวต่อไป) เมื่อเสร็จแล้วจะกลับไปเอดิเตอร์ (เพื่อไปวินโดว์อีดิตคือไม่วนรอบอยู่ภายในดังเช่นในวินโดว์อีดิต)

รหัสคำสั่ง

เมื่อผู้ใช้ทำงานกับเอดิเตอร์ (เช่นเดียวกับที่เราทำงานกับเทอร์โบปาสคาลครับ) ผู้ใช้จะสั่งการผ่านคีย์คอนโทรลหนึ่งคีย์บ้าง สองคีย์บ้าง หรือคีย์หน้าที่พิเศษ ทำให้ยากต่อการทำงาน ผมจึงกำหนดขึ้นเป็นรหัสคำสั่งดังแสดงในรูปที่ 1 และมอบให้โปรแกรมย่อย GetKey เป็นผู้แปลคีย์ที่ผู้ใช้กดเป็นรหัสคำสั่ง รหัสเหล่านี้แบ่งออกได้เป็น 3 กลุ่มใหญ่คือ

- วินโดว์อีดิต ตั้งแต่ nop ถึง actok
- เท็กซ์อีดิต ตั้งแต่ chnghi ถึง chngdir
- เอดิเตอร์ คือที่เหลือ

ซึ่งทั้งหมดนี้มีทั้งที่ GetKey แปลให้คือ ตั้งแต่ curlft ถึง funckey (มีเว้นอยู่บ้าง) และที่วินโดว์อีดิตเปลี่ยนรหัสให้ใหม่คือ ตั้งแต่ newpge คือ newonelne และที่ได้จากการที่ผู้ใช้เลือกรายการในเมนูคือ chngdir, import, export, load save, dir, chngdir, exito, xclose และ quit นอกนั้นเช่น help ถึง ff7 และ user เตรียมไว้สำหรับการใช้งานในอนาคต

วิเคราะห์รหัสคำสั่ง

รหัสคำสั่งเหล่านี้ผมนำมาจากการใช้งานเทอร์โบปาสคาลเท่าที่จำได้ เพราะต้องการให้โปรแกรมนี้เป็นโปรแกรม

ขนาดเล็ก มีสิ่งอำนวยความสะดวกเท่าที่จำเป็นเท่านั้น

เมื่อผมบันทึกรายการคำสั่งที่ต้องการโดยเขียนเป็นรหัสคำสั่งแล้ว ได้นำมาจัดลำดับใหม่ตามการแบ่งมอบงานเป็น 3 กลุ่มดังกล่าวมา (และยังแบ่งย่อยต่อไปอีก) ทำอยู่หลายครั้งครับจึงได้รหัสคำสั่งตามรูปที่ 1 เสร็จแล้วจึงนำมาวิเคราะห์เป็นงานป้อนข้อมูลกับงานอื่นเพื่อหาทางเดินของโปรแกรมในการเขียนโปรแกรมต่อไป คือ

```
TYPE

  CommCode =
   (nop,curlft,currgt,bckspce,tab,lftwrd,rgtwrd,
    lftrow,rgtrow,delch,delEOL,togblk,insrt,lneok,
    curup,curdwn,toppge,botpge,blkmrk,actok,chnghi,
    wndup,wnddwn,delLne,insLne,newLne,reenter,
    sttmrk,aftmrk,cpyblk,mveblk,delblk,
    pgeup,pgedwn,tostt,toaft,fstpge,lstpge,
    import,export,newpge,newfrmpst,newonelne,newpst,
    prnblk,srch,replce,schfnd,again,
    fiopr,load,save,readfrm,wrtto,dir,chngdir,
    help,ff4,ff5,ff6,ff7,mvewnd,cycwnd,user,
    ctrlkey,funckey,exito,xclose,reopen,quit,anerr);

  ChrSet = SET OF Char;
```

รูปที่ 1 แสดงการกำหนดรหัสคำสั่ง

งานป้อนข้อมูล

สามารถแบ่งออกตามลักษณะของทางเดินของโปรแกรมได้เป็น 2 ส่วนดังแสดงในรูปที่ 2 กล่าวคือ

1. เป็นงานที่ดำเนินการสำเร็จอยู่ภายในวินโดว์อีดิตเป็นวงรอบไม่ออกมาเลย คือตั้งแต่ curlft ถึง atok เว้น curup กับ curdwn อาจออกไปนอกเพลตได้ ผมจึงให้เปลี่ยนรหัสเป็น wndup กับ wnddwn ตามลำดับ

2. เป็นงานที่เริ่มจากวินโดว์อีดิตไปดำเนินการในเท็กซ์อีดิต แล้วกลับมาจบ (ขึ้นต้น nop ใหม่) ที่วินโดว์อีดิตคือ ตั้งแต่ wndup ถึง lstpge งานส่วนนี้มีทั้งที่เปลี่ยนข้อมูลในเพลตทั้งหมดหรือบางส่วน ผมจึงกำหนดรหัสคำสั่งขึ้นใหม่ 3 รหัส ให้เท็กซ์อีดิตเปลี่ยนเมื่อจะส่งกลับไปทำงานในวินโดว์อีดิต เพื่อการแสดงภาพบนจอภาพ คือ

- newpge ให้เปลี่ยนภาพทั้งเพลต
- newfrmpst ให้เปลี่ยนภาพตั้งแต่เคอร์เซอร์ไปจนถึงบรรทัดสุดท้าย
- newonelne ให้เปลี่ยนภาพในบรรทัดของเคอร์เซอร์บรรทัดเดียว

-รหัสที่ไม่เปลี่ยนคือ curup curdwn และ newlne เพราะมีการดำเนินการเป็นพิเศษ

- รหัสที่จบการทำงานในเท็กซ์อีดิต (แต่ผมจัดรวมไว้ในกลุ่มนี้ด้วย) คือ prnblk และเมื่อดำเนินการส่งข้อความในบล็อกไปออกเครื่องพิมพ์แล้วให้เปลี่ยนรหัสเป็น nop

รหัสคำสั่ง	วินโดว์อีดิต	เอดิเตอร์	เท็กซ์อีดิต	เอดิเตอร์	วินโดว์อีดิต
curlft..lneok	(*)				
curup, curdwn	(*)/wndup, wnddwn				
toppge..actok	(*)				
chnghi	(	x	newpge		
wndup, wnddwn	(	x	*	x	)
delLne..reenter	(	x	newfrmpst	x	)
newLne	(	x	*	x	)
sltmrk, aftmrk	(	x	blkmrk	x	)
cpyblk..delblk	(	x	newpge /newonelne	x	)
pgeup..lstpge	(	x	newpge	x	)
prnblk	(	x	) nop	-	-
srch, replce	ตามคำอธิบาย				

รูปที่ 2 แสดงทางเดินของโปรแกรมตามรหัสคำสั่ง ที่เริ่มจากวินโดว์อีดิต

(	จุดเริ่มต้นของรหัสคำสั่งนี้
*	การปฏิบัติ
)	จบการทำงานตามรหัสคำสั่งนี้
x	ประสานการติดต่อ
ชื่อรหัส	มีการปฏิบัติ และเปลี่ยนรหัสคำสั่งใหม่
/	อย่างใดอย่างหนึ่ง
-	ไม่มีการปฏิบัติ

งานอื่น

คืองานที่ไม่เกี่ยวกับการป้อนข้อความโดยตรง สามารถแบ่งตามทางเดินของโปรแกรมได้เป็น 3 ส่วนคือ แสดงในรูปที่ 3 คือ

1. งานค้นคำ เป็นงานที่เริ่มจากวินโดว์อีดิต ไปทำงานในเท็กซ์อีดิตมีอยู่ 2 คำสั่งคือ srch (ค้นคำ) กับ replce (ค้นและแทนที่คำ) สำหรับ srch จบการทำงานในเท็กซ์อีดิต (เปลี่ยนรหัสเป็น nop) ส่วน replce มีการดำเนินการที่ยอกย้อนพอสมควร จึงขอนำไปกล่าวในเรื่องของมัน

2. งานเมนู งานพวกนี้ถึงแม้จะเริ่มจากวินโดว์อีดิตที่ให้แสดงเมนู แต่ถือว่าเริ่มจากเอดิเตอร์เพราะเอดิเตอร์เป็นผู้เข้ารหัสไปทำงานในเท็กซ์อีดิต ได้แก่ งานส่งข้อความออกหรือรับข้อความเข้า การดำเนินการกับไฟล์และงานเกี่ยวกับคีย์ฟังก์ชัน F1 ถึง F10 คือ ตั้งแต่รหัส import ถึง user

3. งานพิเศษ เป็นรหัสที่เตรียมไว้หากผู้เขียนโปรแกรมเอดิเตอร์ต้องการใช้งาน รวมทั้งการกลับไปคอมพายเลอร์ คือ ตั้งแต่รหัส ctrlkey ถึง anerr

การประสานการติดต่อ

จากทางเดินของโปรแกรมที่แบ่งออกเป็น 2 ฝ่าย มีเอดิเตอร์อยู่ตรงกลางเป็นผู้ประสานการติดต่อนั้น ผมได้จัดให้มีการติดต่อและการประสานการติดต่อดังนี้

ทางรหัสคำสั่ง

คือที่ให้เอดิเตอร์สั่งปฏิบัติ (เรียกใช้กิจกรรม Action) ต่อวินโดว์อีดิตครั้งหนึ่งและต่อเท็กซ์อีดิตครั้งหนึ่งสลับกันไปมา ผมให้ผ่านรหัสคำสั่งไปและขอคำที่เปลี่ยนคืนมาด้วยเป็นการประสานการติดต่อแบบหนึ่ง

รหัสคำสั่ง	เอดิเตอร์	เท็กซ์อีดิต	เอดิเตอร์	วินโดว์อีดิต
import	(	newpge	x	)
export	(	) nop	-	-
load	(	newpge	x	)
save	(	) nop	-	-
reopen	(	newpge	x	)

รูปที่ 3 แสดงทางเดินของโปรแกรมตามรหัสคำสั่ง ที่เริ่มจากเอดิเตอร์ เครื่องหมายเช่นเดียวกับในรูปที่ 1

ทางโปรแกรมย่อย

แต่เพียงรหัสคำสั่งอย่างเดียวไม่เพียงพอจะต้องส่งข้อมูลให้กันและกันด้วย เพื่อให้เป็นการเขียนโปรแกรมในแบบโอโอพีที่ไม่ให้เข้าถึงข้อมูลโดยตรง (แต่ความจริงเพื่อความสะดวกในการเขียนโปรแกรมของผมมากกว่า) จึงให้แต่ละฝ่ายจัดกิจกรรมสำหรับส่งคำและรับคำแก่กันดังนี้

ดังแสดงในรูปที่ 4 ตอนบน
* เมื่อออกจากเท็กซ์อีดิต
- ตำแหน่งเคอร์เซอร์บนเพลต
- ตำแหน่งต้นบล็อกและท้ายบล็อกบนเพลต

ดังแสดงในรูปที่ 4 ตอนล่าง ซึ่งผมจะให้ดำเนินการโดยไม่ตรวจสอบว่ามีการเปลี่ยนแปลงหรือไม่ เพราะเห็นว่าเวลาที่ใช้ตรวจสอบกับเวลาที่ใช้ส่ง-รับค่าน่าจะพอ ๆ กัน

* เมื่อออกจากวินโดว์อีดิต
- ตำแหน่งของเคอร์เซอร์บนเพลต
- ความสูงของหน้าต่าง เฉพาะเมื่อผู้ใช้เปลี่ยนขนาดของหน้าต่าง

โดยตรง

เมื่อผู้ใช้ต้องการค้นคำหรือค้นและแทนคำ เมื่อมาทำงานในเท็กซ์อีดิตแล้วทางเท็กซ์อีดิตต้องการคำที่จะค้น คำที่จะแทน และออปชัน เช่น จะให้ค้นกี่ครั้ง ซึ่งเป็นการติดต่อกับผู้ใช้ทางคีย์บอร์ดและจอภาพ หากจะให้มีการประสานการติดต่อดังกล่าวมาผมเกรงจะไม่สะดวก จึงจัดให้มีการติดต่อโดยตรง โดยทำเป็นตัวแปรโปรแกรมย่อย (procedure variable) ในเท็กซ์อีดิตเพื่อเรียกใช้โปรแกรมย่อยในวินโดว์อีดิตโดยตรง แต่เอดิเตอร์จะต้องเป็นผู้ให้ค่าแอดเดรสของโปรแกรมย่อยในวินโดว์อีดิตแก่ตัวแปรนี้เมื่อเริ่มต้น จึงนำมารวมไว้ในหัวข้อนี้ด้วย

```
procedure  InfoPstn(var row,col : Byte);
procedure  RecWndPstn(row,col : Byte);
procedure  InfoNewHi(var nhi : Byte);
procedure  RecWndNewHi(nhi : Byte);


procedure  InfoBlock(var sttrow,aftrow,sttcol,aftcol : Byte);
procedure  RecTxtBlock(srow,afrow,scol,afcol : Byte);
procedure  InfoCurPstn(var row,col : Byte);
procedure  RecTxtPstn(row,col : Byte);
```

รูปที่ 4 กิจกรรมที่ใช้ในการติดต่อ

ทางเอดิเตอร์

ความจริงการติดต่อทุกแบบผ่านทางเอดิเตอร์ทั้งนั้น แต่กรณีนี้เป็นกรณีพิเศษและไม่ใช่เป็นการผ่านค่าข้อมูล กล่าวคือ เมื่อเท็กซ์อีดิตค้นคำพบต้องการแสดงข้อความในหน้านั้น หากให้ติดต่อผ่านทางรหัสคำสั่งก็จะทำงานเป็นวงรอบอยู่ในวินโดว์อีดิต จึงให้เอดิเตอร์เรียกกิจกรรมที่นำข้อความออกแสดงทางจอภาพแล้วกลับมาโดยตรง

การติดต่อทั้งสามทางนี้เท่านั้นที่เป็นตัวเชื่อมโยงโปรแกรมทั้งสองกลุ่มเข้าด้วยกัน ดังนั้นหากมีการสร้างยูนิตของฝ่ายหนึ่งฝ่ายใดใหม่ เช่น ต้องการเปลี่ยนวินโดว์อีดิตเป็นแบบกราฟิก ก็เพียงแต่ให้คงสายการติดต่อนี้ไว้ ในการนำมาใช้ก็เพียงแต่คอมพายล์ซอร์สโปรแกรมของเอดิเตอร์ใหม่ โดยไม่ต้องยุ่งกับซอร์สโปรแกรมของเท็กซ์อีดิต ก็จะได้เอดิเตอร์ที่เป็นกราฟิก

การแบ่งมอบงานในกลุ่มเท็กซ์อีดิต

จากงานที่แบ่งมอบให้กับกลุ่มเท็กซ์อีดิตและจากการวิเคราะห์รหัสคำสั่ง ผมได้นำมามอบให้แก่สกุลต่าง ๆ ดังนี้คือ

- สกุล Collection ทำหน้าที่ในการเก็บข้อมูลทั้งในการเก็บต่อท้าย แทรกแทนที่ และถอนข้อมูลที่เก็บไว้

- สกุล StrCollect ทำหน้าที่ในการดำเนินการเป็นบล็อก ได้แก่ การลอกหรือย้ายข้อมูลไปยังอีกตำบลที่หนึ่ง การถอน (คือลบหรือยกเลิก) ข้อมูลเป็นบล็อก การทำงานเป็นบรรทัด ได้แก่ การถอน และแทรกบรรทัด การตัดและต่อบรรทัด

```
UNIT  TXTEDIT;

Interface

uses  crt,prime,editroot,collect,strclct;

TYPE

  TextEdit = OBJECT(StrCollect)
     plate : NotePadT;
     topln : Word;
     wndhi,newhi,dhi : Byte;
     currow,curcol : Byte;
     comcde : CommCode;
     exprtflg,closeflg : Boolean;
```

รูปที่ 5

- สกุล TextEdit ดำเนินการในการเตรียมเพลตให้แก่วินโดว์อีดิตอันได้แก่การเตรียมเพลตเมื่อเริ่มต้น เมื่อมีการเลื่อนหน้าขึ้นลงหรือไปยังตำแหน่งที่กำหนด และเป็นผู้สั่งการปฏิบัติตามรหัสคำสั่งไปยัง StrCollect มีโปรแกรมย่อยสำหรับการประสานการติดต่อทางโปรแกรมย่อย

- สกุล PreEdit ดำเนินการในเรื่องที่เกี่ยวกับไฟล์ การพิมพ์ข้อความในบล็อกออกทางเครื่องพิมพ์ การค้นคำและการค้นและแทนที่คำ

เรื่องของสกุล Collection กับ StrCollect ได้กล่าวไปแล้ว ในฉบับนี้จะนำเรื่องของ TextEdit กับ PreEdit มากล่าวถึงต่อไปครับ

ตัวแปรในสกุล TextEdit

ในการกำหนดสกุลนี้ผมได้กำหนดตัวแปรก่อนอีกเช่นกัน ตามงานที่ได้รับมอบมาดังแสดงในรูปที่ 5 คือ

- plate เพื่อเป็นเพลตดังกล่าวมาเป็นข้อมูลแบบพอยน์เตอร์ไปยังอะเรย์ของพอยน์เตอร์ไปยัง String

- topln เพื่อแสดงว่าโรว์แรกของเพลตเป็นไลน์ที่เท่าไรของบัฟฟ์ เป็นค่าลบหนึ่ง เช่น เมื่อเริ่มแรก row ที่ 1 จะตรงกับไลน์ที่ 1 จะได้ค่า topln เป็น 0 (เพื่อเมื่อนำค่าโรว์มาบวกจะได้เป็นค่าไลน์โดยตรง)

- wndhi เพื่อเก็บความสูงของหน้าต่างในขณะนั้น และจะเป็นจำนวนโรว์ของเพลตด้วย

- newhi เมื่อ RecWndNewH รับค่าความสูงของหน้าต่างใหม่มา นำมาเก็บในตัวแปรนี้ก่อน

- dhi เก็บค่าเศษ 1 ส่วน 3 ของความสูงของหน้าต่าง เพื่อให้แสดงตำแหน่งของคำที่พบในหน้าใหม่ประมาณนี้

- currow และ curcol เพื่อเก็บค่าโรว์และคอลัมน์ที่รับมาจากวินโดอีดิต

- concde เพื่อเก็บค่ารหัสคำสั่งในขณะนั้น

- exprtflg เพื่อแสดงสถานะว่าใช้ได้ส่งข้อความออก (ให้แก่เอดิเตออื่น)

- closeflg เพื่อแสดงสถานะของเอดิเตอร์นี้ว่าได้ถูกยกเลิกหรือไม่

กิจกรรมหลักของ TextEdit

ตามเนื้อหาที่กล่าวมาผมได้กล่าวถึงแต่การติดต่อระหว่างเท็กซ์อีดิตกับวินโดว์อีดิตเท่านั้น แต่ยังมีงานอีกจำนวนหนึ่ง ซึ่งเป็นงานที่สัมพันธ์กับเอดิเตอร์โดยตรง และถือว่าเป็นกิจกรรมหลักของสกุลนี้ ดังรูปที่ 6 คือ

```
CONSTRUCTOR  Init(numlne : Word; windowhi : Byte;
                                  var xplate : Pointer);
procedure  Action(var comm : CommCode);
procedure  ExitEdit(var txt : Pointer);
procedure  CloseText;
procedure  NewText;
procedure  PrepPlate;
procedure  GotoNewPstn(lne : Word; chn : Byte;
                                  var code : CommCode);

(* sub-routine for external call and sent only  *)
procedure  RecWndNewHi(nhi : Byte);
procedure  RecWndPstn(row,col : Byte);
procedure  InfoBlock(var sttrow,aftrow,sttcol,aftcol : Byte);
procedure  InfoCurPstn(var row,col : Byte);

DESTRUCTOR   Fin;  VIRTUAL;
```

รูปที่ 6

Init

ดำเนินการเตรียมค่าต่าง ๆ ดังกล่าวมา รวมทั้งเตรียมเพลตส่งกลับไปด้วย (เป็นเพลตว่าง)

Action

เป็นกิจกรรมที่รับคำสั่งจากเอดิเตอร์ทางรหัสคำสั่งมาดำเนินการ ตามรูปที่ 7 คือ

- ดำเนินการตามรหัสดังที่จะกล่าวในหัวข้อ กิจกรรมตามรหัส
- ถ่ายข้อมูลจากบัฟฟ์ให้เพลตทั้งเพลต และยกเลิกหน่วยความจำของไลน์ที่ถ่ายทอดค่า

ExitEdit

ถ่ายข้อมูลในเพลตให้แก่บัฟฟ์ และให้ค่า buff แก่ txt เมื่อผู้ใช้ต้องการกลับไปคอมพายเลอร์

Closetext

ในกรณีที่ผู้ใช้เปิดเอดิเตอร์เพื่อใช้งานหลาย ๆ งาน หากเกิดปัญหาเรื่องหน่วยความจำไม่พอ ผู้ใช้อาจปิดเอดิเตอร์ตัวใดตัวหนึ่งเพื่อเรียกหน่วยความจำคืนมา ในการนี้เป็นหน้าที่ ของเอดิเตอร์กับผู้ใช้ว่าจะเก็บข้อมูลลบไฟล์หรือไม่ แล้วให้ TextEdit ดำเนินการโดยเคลียร์บัฟฟ์ซึ่งจะยกเลิกหน่วยความจำของไลน์ทุกไลน์ในบัฟฟ์ (แต่ไม่ได้ยกเลิกหน่วยความจำที่เป็นบัฟฟ์โดยตรง) และเคลียร์เพลตคือ ยกเลิกหน่วยความจำของโรว์ทุกโรว์ในเพลต

PrepPlate

เมื่อกลับเข้ามาใหม่จะให้เท็กซ์อีดิตเป็นผู้ตรวจ โดยเรียกใช้ PrepPlate ซึ่งจะตรวจว่าออกไปทาง ExitEdit หรือ CloseText ด้วยการตรวจแฟลก Closeflg และเตรียมเพลตให้ใหม่ หากออกไปทาง CloseText

NewText

เพื่อเซตตำแหน่งของเคอร์เซอร์ในบัฟฟ์ และบล็อกให้เหมือนเมื่อเริ่มต้น (Init)

GatoNewPstn

เพื่อกำหนดตำแหน่งของเคอร์เซอร์ให้ตรงตามที่โปรแกรมภายนอก เช่น คอมพายเลอร์กำหนดมา โดยให้ค่าเคอร์เซอร์และถ่ายข้อมูลจากบัฟฟ์ให้แก่เพลตด้วย

กิจกรรมในการติดต่อของ TextEdit

เพื่อการส่งและรับข้อมูลกับวินโดว์อีดิต มีเอดิเตอร์เป็นผู้ประสานการติดต่อ ดังรูปที่ 6 ได้แก่

RecWndNewHi

ให้ค่า nhi แก่ newhi และคำนวณค่า dhi ใหม่

RecWndPstn

ให้ row แก่ currew และ col แก่ curcol

InfoBlock

นำค่าบล็อกที่เก็บไว้เป็นค่าไลน์ใน sttlne และ aftlne มาแปลงเป็นค่าโรว์ โดยลบด้วย topIne ให้แก่ sttrow และ aftrow ตามลำดับ โดยตรวจดูด้วยว่าถ้าเป็นบล็อกที่ไม่ถูกต้องจะให้ sttrow เป็น 0 หรือถ้าบล็อกอยู่นอกหน้าต่างก็จะจัดค่าให้ตามความเหมาะสม ส่วน

ค่าคอลัมน์จะให้จาก sttchn และ aftchn แก่ sttcal และ aftcal ตามลำดับโดยตรง

InfoCurPstn

นำตำแหน่งเคอร์เซอร์ที่เก็บไว้เป็นไลน์ใน dstlne มาแปลงเป็นค่าโรว์โดยลบด้วย topline ให้แก่ row และให้ค่า dstchn ให้แก่ col โดยตรง

กิจกรรมตามรหัสคำสั่งของ TextEdit

เป็นกิจกรรมเพื่อทำงานตามรหัสคำสั่ง ดังแสดงในรูปที่ 8 ขอให้ดูเปรียบเทียบกับรูปที่ 7 (กิจกรรมใดที่ไม่กล่าวในหัวข้อนี้เป็นกิจกรรมที่กำหนดไว้ในสกุล StrCollect) ที่อยู่ในสกุลนี้ ได้แก่

WindowUp

เพื่อเลื่อนหน้าต่างขึ้นบน 1 บรรทัด โดยลดค่า topline ลงหนึ่งหากมีค่ามากกว่าศูนย์ ดังนั้นเมื่อถ่ายค่าจากบัฟฟ์ให้เพลตข้อความที่ได้ไปจะเป็นข้อความที่เริ่มจากเหนือบรรทัดเดิม 1 บรรทัด เมื่อนำออกแสดงจะเห็นเหมือนเป็นการเลื่อนหน้าต่างขึ้นบน แต่ถ้า topline เป็นศูนย์คือ เลื่อนหน้าต่างไม่ได้ จะเปลี่ยนค่ารหัสเป็น nop เพื่อแจ้งแก่วินโดว์อีดิตว่าไม่ต้องดำเนินการใด ไม่เช่นนั้นวินโดว์อีดิตจะแสดงภาพในหน้าต่างใหม่ทั้งหมด

WindowDown

เพื่อเลื่อนหน้าต่างลงล่าง 1 บรรทัด ดำเนินการเช่นเดียวกับ WindowUp แต่เป็นการเลื่อนลงล่างไม่ให้เกินไลน์ที่ lastPstn

DeleteLine

InsertLine

เพื่อการลบและการแทรกบรรทัดด้วยการเรียกใช้กิจกรรมชื่อเดียวกันใน StrCollect ที่ต้องกำหนดกิจกรรมนี้ทับเพื่อการให้ค่า dstchn เป็น 1 และเปลี่ยนค่ารหัสเป็น newfrmpst เพื่อให้วินโดว์อีดิตแสดงภาพใหม่ตั้งแต่โรว์ที่ตำแหน่งเคอร์เซอร์เป็นต้นไป (คือไม่ต้องแสดงใหม่ทั้งหน้าต่าง)

```
PROCEDURE  TextEdit.Action(var comm : CommCode);
BEGIN
  comcde := comm;

  if  comcde = sttmrk  then
    begin  SetStartBlock(topline+currow,curcol);
           comcde := blkmrk  end
  else if  comcde = aftmrk  then
    begin  SetAftBlock(topline+currow,curcol);
           comcde := blkmrk  end
  else
    begin
    PlateToBuff;
    SetCurPstn(topline+currow,curcol);
    CASE  comcde  OF
    wndup   : WindowUp;
    wnddwn  : WindowDown;
    delline : DeLeteLine;
    insline : InsertLine;
    newline : DoEnter;
    reenter : ReEnterLine;
    cpyblk  : BlockCopy;
    mveblk  : BlockMove;
    delblk  : BlockDelete;
    pgeup   : PageUp;
    pgedwn  : PageDown;
    tostt   : ToDestination(sttline,sttchn);
    toaft   : ToDestination(aftline,aftchn);
    fstpge  : ToDestination(1,1);
    lstpge  : ToDestination(lastPstn,1);
    chnghi  : ChangeWndHi;
    import  : DoImport;
    export  : DoExport;
    END;  { case }
    BuffToPlate;
    end;

  if  comcde in [cpyblk..delblk]  then
    comcde := newpge;
  comm := comcde;

  CheckMem(0);
  if  break  then  begin  PlateToBuff;  Exit  end;

END;
```

รูปที่ 7

```
(*  command code sub-routine  *)
   procedure  WindowUp;
   procedure  WindowDown;
   procedure  DeleteLine;
   procedure  InsertLine;
   procedure  ReEnterLine;
   procedure  PageUp;
   procedure  PageDown;
   procedure  ChangeWndHi;
   procedure  DoImport;
   procedure  DoExport;
```

รูปที่ 8

ReEnterLine

เพื่อการนำข้อความบรรทัดล่างไปต่อท้ายข้อความในบรรทัดบน (คือ เมื่อผู้ใช้กดคีย์ back space ที่ต้นบรรทัด) โดยตรวจว่าถ้าขณะนั้นอยู่ที่บรรทัดบนสุดให้เลื่อนหน้าต่างขึ้นบน 1 บรรทัด ถ้าเลื่อนไม่ได้ให้กลับ นอกนั้นให้เรียกใช้กิจกรรม DoReEnter ใน StrCollect แล้วให้รหัสเป็น newfrmpst

PageUp

PageDown

เลื่อนหน้าต่างขึ้นบนหรือลงล่าง 1 หน้าต่าง โดยลบหรือบวกค่า toplne ด้วยค่าความสูงของหน้าต่างลบด้วย 1 แต่ค่า toplne ต้องไม่น้อยกว่าศูนย์ แล้วให้ค่ารหัสเป็น newpge เพื่อให้วินโดว์อีดิตแสดงภาพทั้งหน้าต่าง

ChangeWndHi

เปลี่ยนความสูงของเพลตใหม่โดยยกเลิกสตริงในเพลตเดิมทั้งหมดแล้วกำหนดใหม่ให้เท่ากับความสูง และเปลี่ยนรหัสเป็น newpage

DoExport

เพื่อการส่งข้อความออกโดย ให้ค่า buf แก่ xbuff ค่า sltlne กับ xsttlne ค่า aftlne กับ xaltlne และให้ exprtflg เป็นจริง

ตัวแปรที่ขึ้นต้นด้วยอักษร x คือตัวแปรที่ผมกำหนดไว้เป็นส่วนกลางในยูนิต ดังนั้นการย้ายข้อความจากเอดิเตอร์หนึ่งไปยังอีกเอดิเตอร์หนึ่ง (ซึ่งผู้ใช้จะมองเห็นเหมือนจากหน้าต่างหนึ่งไปยังอีกหน้าต่างหนึ่ง) ทำได้ครั้งละคู่เท่านั้น

```
(*  Internal sub-routine  *)
    procedure  DoPrepPlate;
    procedure  FreePlate;
    procedure  ClearPlate;
    procedure  AddInsLne(lne : Word; sptr : StrPtrT);
    procedure  BuffToPlate;
    procedure  PlateToBuff;
    procedure  BtoPlne(lne : Word; row : Byte);
    procedure  PtoBlne(lne : Word; row : Byte);
    procedure  ClrPltRow(row : Byte);
    procedure  ClearBuff;
    procedure  ToDestination(lne : Word; chn : Byte);

    procedure  DispBuff;
    procedure  DispPlate;
           END;
```

รูปที่ 9

```
UNIT  PREEDIT;

Interface
uses  crt,prime,editroot,collect,strclct,dos,txtedit,stropt;

TYPE
     Str41    = String[41];
     Str7     = String[7];
     StrOptProc = procedure(var mssge,ans : Str79; opt : ChrSet);

    PreEditor = OBJECT(TextEdit)
        objfi  : Text;
        finame,scndname : Str79;
        cdecom,lstcom,dispcde : CommCode;
        sveok : Boolean;
        srchstr,rplcstr : Str41;
        option : Str7;
        srchlen,rplclen : Byte;
        lne : Word;
        chn,cnt1,cnt2 : Byte;
        bckwrd,upr : Boolean;

        callGetAnsStrOpt : StrOptProc;
```

รูปที่ 10

โปรแกรมประกอบ

โปรแกรมเพื่อสนับสนุนการทำงานของกิจกรรมดังกล่าวมาอยู่ในรูปที่ 9 เนื่องจากเป็นโปรแกรมง่าย ๆ และชื่อคงบอกหน้าที่อยู่แล้วจึงของดอธิบายครับ

ข้อมูลในสกุล PreEdit

เนื่องจาก PreEdit ทำงานเกี่ยวกับไฟล์และการค้นหาแทนที่คำ จึงมีตัวแปรด่วนข้างมาก ดังแสดงในรูปที่ 10 คือ

- objfi เป็นเท็กซ์ไฟล์ของเอดิเตอร์นี้ ขึ้นอยู่กับเราจะ adsigne ให้เป็นไฟล์ในดิสก์ หรือเป็นเครื่องพิมพ์ เพื่อให้โปรแกรมในการอ่านและเขียนกับไฟล์ดำเนินการ

- finame สำหรับเก็บชื่อไฟล์ (รวมทั้ง path) ของเอดิเตอร์นี้ ในการโหลดและเซฟข้อมูลทั้งบัฟฟ์

- scndname สำหรับเก็บชื่อไฟล์ (รวมทั้ง path) ในการอ่านและเขียนเป็นบล็อก

- edecom เก็บรหัสคำสั่งขณะนั้น

- lstcom เก็บรหัสคำสั่งครั้งหลังสุด

- sveok เป็นแฟลกแสดงสถานะว่าได้เก็บข้อมูลในบัฟฟ์ลงไฟล์แล้ว

- srchstr เก็บข้อความที่จะค้นหา

- rplcstr เก็บข้อความที่จะนำไปแทนที่

- option เก็บออปชันที่ผู้ใช้เลือก เช่น bu10 (แต่เตรียมโปรแกรมไว้ให้สำหรับจำนวนครั้งเท่านั้น)

- srchlen เก็บความยาวของข้อความที่จะค้นหา

- rplclen เก็บความยาวของข้อความที่จะนำไปแทนที่

- cntl เก็บจำนวนครั้งที่ผู้ใช้ต้องการให้ค้น

- bckwrd เก็บสถานะว่าผู้ใช้ต้องการให้ค้นย้อนหลัง

- upr เก็บสถานะว่าอักษรเล็กเท่ากับอักษรใหญ่

- lne และ chn เก็บตำแหน่งของเคอร์เซอร์ในการค้นครั้งหนึ่ง ๆ (เช่น ผู้ใช้ให้ดำเนินการซ้ำ)

- cat2 เก็บค่าของ cntl ในการค้นครั้งหนึ่ง ๆ

- dispcde เก็บรหัสคำสั่ง ซึ่งเอดิเตอร์จะต้องผ่านไปให้แก้วินโดว์อีดิตเมื่อให้แสดงภาพในการค้นแทนคำ

กิจกรรมหลักของ PreEditor

กิจกรรมหลักของ PreEditor มีเพียง 2 กิจกรรมเท่านั้น แสดงในรูปที่ 11 คือ

Init

เพื่อการเตรียมค่าเริ่มต้น ดังแสดงในรูปที่ 12

Action

เพื่อการปฏิบัติตามรหัสคำสั่ง ดังแสดงในรูปที่ 13 ขอให้สังเกตด้วยว่าถ้ารหัสคำสั่งเป็น load จะไม่ SetCur Pstn คือ จะไม่ให้เคอร์เซอร์ของบัฟฟ์ตรงกับเคอร์เซอร์ของเพลต ทั้งนี้เพราะถ้าเป็นไฟล์เดิมจะต้องอยู่ที่ตำแหน่งเดิม ไม่ใช่ตำแหน่งใหม่

กิจกรรมของรหัสคำสั่งใน PreEditor

กิจกรรมของรหัสคำสั่งใน Pre Editor แสดงไว้ในรูปที่ 11 เช่นกัน กล่าวคือ

LoadFile

โหลดไฟล์ลงหน่วยความจำ โดยมีขั้นตอนการดำเนินการ ดังนี้

- ตรวจแฟลก sveok หากมีไฟล์เก่าค้างอยู่ให้แจ้งชื่อและถามผู้ใช้ว่าจะเซฟหรือไม่ ถ้าเซฟให้เซฟไฟล์เดิม

- ให้ผู้ใช้ป้อนชื่อไฟล์ โดยแจ้งชื่อไฟล์เก่าให้ทราบด้วย

- หากเป็นไฟล์ใหม่ ให้เซตเคอร์เซอร์และบล็อกเหมือนเมื่อเริ่มต้น (Init) โดยเรียกใช้ NewText ใน TextEdit

- เคลียร์บัฟฟ์ (ยกเลิกหน่วยความจำที่เป็นไลน์ในบัฟฟ์)

- ให้โหลดไฟล์ด้วย DoLoad

- ให้รหัสคำสั่งเป็น newpge

นั่นคือ หากผู้ใช้ออกจากเอดิเตอร์

```
CONSTRUCTOR  Init(numlne : Word; wndowhi : Byte;
                  var xplate : Pointer);
procedure  Action(var comm : CommCode);
procedure  LoadFile;
procedure  SaveFile;
procedure  ReadFrom;
procedure  WriteTo;
procedure  PrintBlock;
procedure  SearchString;
procedure  SearchAndReplace;
procedure  DoAgain;
```

รูปที่ 11

```
CONSTRUCTOR  PreEditor.Init(numlne : Word; wndowhi : Byte; var xplate : Pointer
;
BEGIN
  TextEdit.Init(numlne,wndowhi,xplate);
  finame := 'NONAME.TXT';  scndname := '';
  sveok := True;  srchstr := '';  rplcstr := '';
  option := '';  lstcom := nop;
END;
```

รูปที่ 12

ด้วย xclose เมื่อกลับเข้ามาใหม่ เอดิเตอร์จะเป็นผู้ตรวจด้วย Prepplate ใน TextEdit ซึ่งจะเตรียมเพลตให้ ดังนั้นการดำเนินการในขั้นโหลดไฟล์จึงเป็นไปตามปกติ

SaveFile

เก็บข้อมูลในบัฟฟ์ลงไฟล์ โดยตรวจดูถ้าเป็นไฟล์ชื่อ NONAME.TXT จะให้โอกาสผู้ใช้ป้อนชื่อไฟล์ใหม่

ReadFrem

อ่านข้อมูลจากไฟล์ลงบัฟฟ์ในฐานะบล็อก โดยแสดงชื่อไฟล์ใน scndname เพื่อให้ผู้ใช้ป้อนชื่อไฟล์ใหม่

WriteTo

เก็บข้อมูลในบล็อกลงไฟล์ โดยแสดงชื่อไฟล์ใน scndname เพื่อให้ผู้ใช้ป้อนชื่อไฟล์ใหม่เช่นกัน

หมายเหตุ

1. การดำเนินการอ่านและเขียนกับไฟล์ทั้งสี่เรื่องนี้ จะแสดงชื่อไฟล์เดิมเพื่อให้ผู้ใช้เลือก ชื่อเดิม หรือป้อนชื่อใหม่ หรือยกเลิก (กดคีย์ Esc) หากเป็นการยกเลิกจะไม่ดำเนินการใดตัว

2. เช่นเดียวกันในการค้นและค้นแทนที่คำ เมื่อให้ผู้ใช้ป้อนข้อความที่จะเป็นคำที่จะค้น คำสำหรับแทนที่ หรือออปชัน จะให้แสดงข้อความเดิมเพื่อให้ผู้ใช้เลือกดังกล่าวมา หากผู้ใช้ยกเลิกในขั้นตอนใดก็จะไม่ดำเนินการใดต่อเช่นกัน

PrintBlack

พิมพ์ข้อความในบล็อกออกทางเครื่องพิมพ์โดย

- Assign ให้ objji เป็น PRN

```
PROCEDURE  PreEditor.Action(var comm : CommCode);
BEGIN
  IF  comm > newpst  THEN
    BEGIN
    cdecom := comm;
    PlateToBuff;
    if  cdecom <> load  then
      SetCurPstn(toplne+currow,curcol);
    CASE  cdecom  OF
     load     : LoadFile;
     save     : SaveFile;
     readfrm  : ReadFrom;
     wrtto    : WriteTo;
     prnblk   : PrintBlock;
     srch     : SearchString;
     replce   : SearchAndReplace;
     schfnd   : DoSrchRep;
     again    : DoAgain;
    END;  { case }
    BuffToPlate;
    comm := cdecom;
    END
  ELSE
    TextEdit.Action(comm);
END;
```

รูปที่ 13

- ตรวจสอบเครื่องพิมพ์ว่าพร้อมหรือไม่ด้วย PrinterOn หากไม่พร้อมจะถามผู้ใช้ว่าจะยกเลิกหรือจะดำเนินการต่อ (เพื่อให้โอกาสผู้ใช้เปิดเครื่องพิมพ์)

- หากให้ดำเนินการต่อให้พิมพ์ด้วย DoWriteFile

SearchString

ค้นคำเริ่มจากตำแหน่งของเคอร์เซอร์ไปทางท้ายบัฟเฟอร์ โดยเก็บรหัสนี้ไว้ใน lstcom ให้ผู้ใช้ป้อนคำที่จะค้นและออปชัน (ตามที่กล่าวมา) แล้วให้ดำเนินการค้นด้วย DoSearch

SearchAndReplace

ค้นและแทนที่คำจากตำแหน่งของเคอร์เซอร์ไปทางท้ายบัฟเฟอร์ โดยดำเนินการเช่นเดียวกับ SearchString แต่ผู้ใช้ป้อนคำที่จะนำมาแทนที่ด้วย และให้ดำเนินการด้วย StartSrchRep

DoAgain

ดำเนินการค้นคำ หรือค้นแทนที่คำซ้ำอีกครั้ง ขึ้นอยู่กับรหัสคำสั่งเดิมที่เก็บอยู่ใน lstcom จะเป็นอะไร หากเป็นการค้นคำก็จะให้ DoSearch ดำเนินการ หากเป็นการค้นแทรกคำจะให้ Start SrchRep ดำเนินการ

โปรแกรมสนับสนุนของ PreEdit

ได้จัดโปรแกรมสนับสนุนการทำงานของกิจกรรมดังกล่าวมาไว้ในรูปที่ 14 จะขอนำโปรแกรมที่สำคัญมากล่าวถึง คือ

DoLoad

เป็นผู้โหลดข้อมูลจากไฟล์ลงบัฟฟ์ โดย อ่านข้อมูลจากไฟล์ทีละ 1 สตริง นำไปให้แก่บัฟฟ์เริ่มจากไลน์ที่ lne ดังนั้น

```
DESTRUCTOR  Fin; VIRTUAL;
function    GetFileName(var tstnum : Word) : Boolean;
procedure   DoLoad(var nlne : Word);
procedure   DoWriteFile(frmlne,toolne : Word; frmchn,toochn : Byte);
function    PrinterOn : Boolean;
function    LineSearch(sstr : StrPtrT; var stt,aft : Byte) : Boolean;
function    GetSrchStr : Boolean;
function    GetRepStr : Boolean;
function    GetSrchOpt : Boolean;
procedure   DoSearch;
procedure   SentSrchNotFound;
function    Searcher : Boolean;
procedure   StartSrchRep;
procedure   DoSrchRep;
procedure   ReplaceStr(lne : Word; stt,aft : Byte; rstr : Str41);

              END;
```

รูปที่ 14

จึงใช้ได้กับทั้ง LoadFile และ Read-From

DoWriteFile

เก็บข้อมูลในบัฟฟ์เป็นบล็อกลงไฟล์ โดย

- เก็บไลน์แรก หากเป็นเศษจะเก็บเศษลงไฟล์
- เก็บไลน์ที่เหลือเว้นไลน์สุดท้าย
- เก็บไลน์สุดท้ายหากเป็นเศษให้เก็บเศษลงไฟล์ จึงใช้ได้กับทั้ง SaveFile และ WriteTo

callGetAnsStrOpt

เป็นตัวแปรโปรแกรมย่อยอยู่ในรูปที่ 10 (ขอให้ดูการกำหนดแบบข้อมูล StrOptProc ด้วยครับ) เพื่อการเรียกใช้โปรแกรมย่อย GetAnsStrOpt ในวินโดว์อีดิต (ที่เรียกเป็นการติดต่อโดยตรง) โดยผ่านข้อความที่จะให้นำออกแสดงทาง mssge และผ่านข้อความเดิมถ้ามีทาง ans สำหรับ opt หากให้เป็นเซตว่าง GetAnsStrOpt จะอ่านค่าเป็นสตริงให้ ans ถ้าให้มีอักขระในเซตจะตรวจจับคีย์เฉพาะที่บ่งถึงในเซตเท่านั้น และตรวจจับเพียงคีย์เดียวโดยไม่คอยคีย์ Enter ให้คำกลับมาใน res เช่นกัน

โดย GetAnst10pt จะแสดงข้อความและรอรับคีย์ที่ผู้ใช้จะป้อนบนโรว์ที่ 2 ของจอภาพ และถ้าในเซตมีอักขระ Esc อยู่ด้วย จะแสดงเคอร์เซอร์ ณ ตำแหน่งเคอร์เซอร์เดิมสลับกับที่ท้ายข้อความบนโรว์ที่ 2 (คือ เมื่อตรวจพบคำและถามผู้ใช้ว่าจะให้แทนคำหรือไม่)

GetSrchStr

GetRepStr

GetSrchOpt

เป็นฟังก์ชันให้ผู้ใช้ป้อนคำที่จะค้นคำที่จะแทนและออปชันตามลำดับ โดยเรียกใช้ callGetAnsStrOpt ค่าที่ได้มาจะนำไปเก็บในตัวแปร srchstr, rplcstr และ bckwrd กับ upr กับ cntl ตามลำดับ ถ้าผู้ใช้กดคีย์ Esc จะให้ตนเองเป็นเท็จ

LineSearch

เป็นฟังก์ชันค้นคำใน srchstr ว่ามีอยู่ในสตริง sstr หรือไม่ โดยเริ่มต้นค้นตั้งแต่อักขระที่ aft เป็นต้นไป หากพบจะให้ตนเองเป็นจริง และให้ตำแหน่งต้นคำแก่ stt ท้ายคำแก่ aft ถ้าไม่พบจะให้ตนเองเป็นเท็จ

Searcher

ค้น srchstr ในบัฟฟ์ 1 เที่ยว (คือ จบพบหรือไม่พบ) เริ่มจากไลน์ที่ lne และอักขระที่ chn โดยเรียกใช้ LineSearch เมื่อขึ้นไลน์ใหม่จะให้ chn เป็น 1

DoSearch

ค้นหาคำเป็นจำนวนครั้งเท่าที่ผู้ใช้กำหนดใน cntl โดย ให้ตำแหน่งเคอร์เซอร์ในบัฟฟ์แก่ lne และ chn แล้วเรียกใช้ Searcher เป็นจำนวนครั้งใน cntl ถ้าพบจะแสดงข้อความ ณ ตำแหน่งที่พบครั้งสุดท้ายด้วย ToDestination (ค่าใน cntl ไม่เปลี่ยนดังนั้นเมื่อผู้ใช้ให้ตรวจซ้ำจะตรวจซ้ำเป็นจำนวนครั้งเท่าเดิม) ถ้าไม่พบหรือพบน้อยกว่าจำนวนครั้งที่กำหนดจะแสดงข้อความว่าค้นไม่พบ

StartSrchRep

เริ่มต้นการค้นหาและแทนคำ (เพื่อให้ DoAgain สามารถเรียกเข้ามาได้ด้วย) โดยจะให้ค่าเคอร์เซอร์ในบัฟฟ์

แก่ lne และ chn ให้ srchan เป็นจริง แล้วเรียกใช้ DoSrchRep

DoSrchRep

ดำเนินการค้นหาและแทนคำเป็นจำนวนครั้งตามที่ผู้ใช้กำหนด ดังแสดงในรูปที่ 15 จะเห็นว่าโปรแกรมนี้แบ่งออกเป็น 2 ท่อน ท่อนแรกเป็นการค้นคำส่วนท่อนหลังเป็นการแทนคำ คือ

* ท่อนแรก จะทำงานในท่อนนี้เมื่อ srchon เป็นจริง เช่น ในครั้งแรกซึ่งจะ

- ค้นหาคำ 1 ครั้งด้วย Searcher

- หากพบจะให้เตรียมเพลตด้วย ToDestination และให้ srchon เป็นเท็จ ให้รหัสคำสั่งเป็น schfnd ซึ่งเมื่อกลับไปเอดิเตอร์จะเรียกใช้โปรแกรมย่อยในวินโดว์นำภาพในเพลตออกแสดงแล้วกลับมาเท็กซ์อีดิตใหม่ ซึ่งจะเข้ามาทำงานใน DoSrchRep โดยตรง (ซึ่งจะเป็นการทำงานในท่อนที่สอง)

- หากไม่พบให้แจ้งว่าตรวจไม่พบและจบการค้นหา

* ท่อนหลัง ทำงานในท่อนนี้เมื่อ srchon เป็นเท็จคือ เมื่อกลับมาจากการไปแสดงภาพ ซึ่งจะ

- ถามผู้ใช้ว่าจะให้แทนคำหรือไม่ด้วย callGet AnsStrOpt

- หากผู้ใช้ให้แทนคำ จะแทนคำใน buff ด้วย ReplaceStr ให้ despcde เท่ากับ newonelne เพื่อให้วินโดว์อีดิตเปลี่ยนข้อความบรรทัดนี้บนจอภาพ และให้ srchon เป็นจริง ดังนั้นการทำงานในเที่ยวต่อไปจะไปทำงานในท่อนแรก

- หากผู้ใช้ไม่ได้ให้แทนคำ จะให้ dispcde เป็น actok (ไม่ต้องให้วินโดว์อีดิตแสดงภาพ) แล้วให้ srchon เป็นจริงเช่นกัน

```
PROCEDURE   PreEditor.DoSrchRep;
BEGIN
  if  srchon  then
    if  Searcher  then
      begin
      lstlne := lne;  lstchn := chn;  ToDestination(lne,chn);
      cdecom := schfnd;  srchon := False;
      end
    else  begin  SentSrchNotFound;  cdecom := nop  end

  else {  if  cdecom = schfnd  then  }
    begin
    mssge := 'Replace (Y/N)?';  ans := '';
    GetAnsStrOpt(mssge,ans,['Y','N',esc]);
    if  ans[1] = esc  then  begin  cdecom := nop;  Exit  end;
    if  ans[1] = 'Y'  then
      begin  ReplaceStr(lne,chn-srchlen,chn,rplcstr);
             comcde := newonelne  end
    else  comcde := actok;
    if  cnt2 > 1  then  dec(cnt2)  else  cdecom := actok;
    srchon := True;
    end;
END;
```

รูปที่ 15

- แต่ถ้าครบจำนวนครั้งที่กำหนดแล้วจะให้รหัสคำสั่งเป็น actok เพื่อยุติการทำงานในวงรอบต่อไป

ขออธิบายเสริมด้วยครับคือ ในเอดิเตอร์เมื่อพบรหัสคำสั่ง schfnd จะต้องตรวจคำใน dispcde หากเป็น newpge หรือ newonelne จึงจะให้วินโดว์อีดิตแสดงภาพใหม่ แต่ถ้าเป็น actok ไม่ต้องให้แสดงภาพ และจะทำงานเป็นวงรอบตามที่กล่าวมา จนรหัสคำสั่งเป็น actok (ครบจำนวนครั้ง)

ที่ผมต้องเขียนโปรแกรมวุ่นวายเช่นนี้ เพราะโปรแกรมแยกกันอยู่ครับ

ซอฟต์เมล์

ก็เป็นอันว่าจบสำหรับทางด้านเท็กซ์อีดิต ผมได้ให้โปรแกรมทั้งหมดรวมทั้งโปรแกรมทดสอบแก่ซอฟต์เมล์แต่ถ้าไม่รีบร้อนผมคิดว่ายังไม่ต้องจัดหาครับ เพราะในฉบับหน้าผมคาดคิดว่าจะให้โปรแกรมทั้งหมดได้ สำหรับการใช้งานคงจะล้าสมัยไปแล้ว เพราะบอร์แลนด์ได้ให้โปรแกรมประเภทเดียวกันกับที่ผมกล่าวถึงนี้ (แต่วิจิตรกว่ากันมาก) มากับเทอร์โบปาสคาล 6.0 ส่วนในด้านการศึกษายังไม่ล้าสมัยครับ